TCS RIO 125 Internship project

Automated Sentiment Analysis of textual comments and feedback

Refrance Links from tcs

1. https://www.intel.com/content/www/us/en/developer/topic-technology/artificial-intelligence/training/courses.html
2. https://python-course.eu/machine-learning/
3. https://scikit-learn.org/stable/tutorial/index.html
4. https://pandas.pydata.org/pandas-docs/version/0.15/tutorials.html
5. https://www.tensorflow.org/tutorials
6. https://www.tensorflow.org/tutorials/keras/classification
7. https://www.tensorflow.org/tutorials/images/cnn
8. https://www.tensorflow.org/text/tutorials/text_classification_rnn
9. https://www.intel.com/content/www/us/en/developer/learn/course-deep-learning.html
10. https://www.intel.com/content/www/us/en/developer/learn/course-machine-learning.html

Importing libraries

```
import numpy as np
import pandas as pd
print('done')
```

```
done
```

Importing dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
%cd /content/drive/MyDrive/Msc AI /TCS iON Rio -125 internship
!ls
```

```
/content/drive/MyDrive/Msc AI /TCS iON Rio -125 internship
 a1_RestaurantReviews_HistoricDump.csv  'Sentiment Analysis P1.ipynb'
 a2_RestaurantReviews_FreshDump.csv
```

```
dataset = pd.read_csv('/content/drive/MyDrive/Msc AI /TCS iON Rio -125 int
```

```
dataset.shape
```

```
(900, 2)
```

```
dataset.head(10)
```

	Review	Liked
0	Wow... Loved this place.	1
1	Crust is not good.	0
2	Not tasty and the texture was just nasty.	0
3	Stopped by during the late May bank holiday of...	1
4	The selection on the menu was great and so wer...	1
5	Now I am getting angry and I want my damn pho.	0
6	Honeslty it didn't taste THAT fresh.)	0
7	The potatoes were like rubber and you could te...	0
8	The fries were great too.	1
9	A great touch.	1

```
dataset['Review'].head()
```

```
0                             Wow... Loved this place.
1                                   Crust is not good.
2             Not tasty and the texture was just nasty.
3    Stopped by during the late May bank holiday of...
4    The selection on the menu was great and so wer...
Name: Review, dtype: object
```

```
dataset.shape
```

```
(900, 2)
```

Data Preprocessing

```
import re
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
corpus=[]
```

```python

for i in range(0, 900):
  review = re.sub('[^a-zA-Z]', ' ', dataset['Review'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in set(all_stopwc
  review = ' '.join(review)
  corpus.append(review)
```

```python
corpus
```

```
['wow love place',
 'crust not good',
 'not tasti textur nasti',
 'stop late may bank holiday rick steve recommend love',
 'select menu great price',
 'get angri want damn pho',
 'honeslti tast fresh',
 'potato like rubber could tell made ahead time kept warmer',
 'fri great',
 'great touch',
 'servic prompt',
 'would not go back',
 'cashier care ever say still end wayyy overpr',
 'tri cape cod ravoli chicken cranberri mmmm',
 'disgust pretti sure human hair',
 'shock sign indic cash',
 'highli recommend',
 'waitress littl slow servic',
 'place not worth time let alon vega',
 'not like',
 'burritto blah',
 'food amaz',
 'servic also cute',
 'could care less interior beauti',
 'perform',
 'right red velvet cake ohhh stuff good',
 'name',
 'hole wall great mexican street taco friendli staff',
 'took hour get food tabl restaur food luke warm sever run around like total overwhelm',
 'worst salmon sashimi',
 'also combo like burger fri beer decent deal',
 'like final blow',
 'found place accid could not happier',
 'seem like good quick place grab bite familiar pub food favor look elsewher',
 'overal like place lot',
 'redeem qualiti restaur inexpens',
 'ampl portion good price',
 'poor servic waiter made feel like stupid everi time came tabl',
 'first visit hiro delight',
 'servic suck',
 'shrimp tender moist',
 'not deal good enough would drag establish',
 'hard judg whether side good gross melt styrofoam want eat fear get sick',
 'posit note server attent provid great servic',
 'frozen puck disgust worst peopl behind regist',
 'thing like prime rib dessert section',
 'bad food damn gener',
 'burger good beef cook right',
 'want sandwich go firehous',
 'side greek salad greek dress tasti pita hummu refresh',
 'order duck rare pink tender insid nice char outsid',
 'came run us realiz husband left sunglass tabl',
 'chow mein good',
 'horribl attitud toward custom talk one custom enjoy food',
```

```
 'portion huge',
 'love friendli server great food wonder imagin menu',
 'heart attack grill downtown vega absolut flat line excus restaur',
```

Data transformation

```python
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1420)
```

```python
X = cv.fit_transform(corpus).toarray()
y = dataset.iloc[:, -1].values
```

```python
X
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```python
y
```

```
array([1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1,
       1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0,
       0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1,
       1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1,
       0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1])
```

```python
type(y)
```

```
numpy.ndarray
```

```python
# Saving BoW dictionary to later use in prediction
import pickle
bow_path = 'c1_BoW_Sentiment_Model.pkl'
pickle.dump(cv, open(bow_path, "wb"))
```

Dividing dataset into training and test set

[ ] ↳ *5 cells hidden*

Model fitting

SVM

```python
# SVM model
#Import svm model
from sklearn import svm
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel
#Train the model using the training sets
clf.fit(X_train, y_train)
#Predict the response for test dataset
Y_pred = clf.predict(X_test)
```

```python
print(Y_pred)
```

```
[1 1 0 1 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0
 1 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1
 1 0 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 0 0
 0 0 1 0 1 1 0 1 1 1 1 0 0 0 1 1 1 0 1 0 1 1 1 0 0 0 0 1 0 1 1 1 1 1 0 1 1
 1 1 0 1 1 1 0 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0]
```

```python
Y_pred.shape
```

```
(180,)
```

```python
import joblib
joblib.dump(clf, 'Clf_Sentiment_Model_SVM')
```

```
['Clf_Sentiment_Model_SVM']
```

```python
type(Y_pred)
```

```
numpy.ndarray
```

```python
df = pd.DataFrame(Y_pred, columns = ['Polarity'])
df.head(6)
```

	Polarity
0	1
1	1
2	0
3	1
4	0
5	0

```python
df.shape
```

```
(180, 1)
```

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Polarity  180 non-null    int64
dtypes: int64(1)
memory usage: 1.5 KB
```

```python
df.count()
```

```
Polarity    180
dtype: int64
```

```python
# Exporting SVM to later use in prediction
import joblib
joblib.dump(clf, 'Clf_Sentiment_Model')
```

```
['Clf_Sentiment_Model']
```

Naive Bays

```python
# importing Naive Bays model from scikit-learn
from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
```

```python
X_train.shape
```

```
(720, 1420)
```

```python
y_train.shape
```

```
(720,)
```

```python
len(y_pred)
```

```
180
```

```python
# Exporting NB Classifier to later use in prediction
import joblib
joblib.dump(classifier, 'c2_Classifier_Sentiment_Model')
```

```
['c2_Classifier_Sentiment_Model']
```

Model performance

```python
type(y_test)
```

```
numpy.ndarray
```

```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn.metrics import confusion_matrix, accuracy_score
# Model Accuracy: how often is the classifier correct?
acc =accuracy_score(y_test, Y_pred)
_cm_ = confusion_matrix(y_test,Y_pred)
print('confusion matrix:\n',_cm_)
print('Accuracy % :',acc*100)
```

```
confusion matrix:
 [[58 20]
 [20 82]]
Accuracy % : 77.77777777777779
```

```python
y_test
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0,
       1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0,
       1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1,
       0, 0, 0, 0])
```

```python
y_pred
```

```
array([0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       1, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 1,
       1, 0, 1, 0])
```

```python
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
# print(cm)
ACc = accuracy_score(y_test, y_pred)
print('Accuracy:',ACc*100)
print('Confusion Matx:\n',cm)
```

```
Accuracy: 73.88888888888889
Confusion Matx:
 [[67 11]
 [36 66]]
```

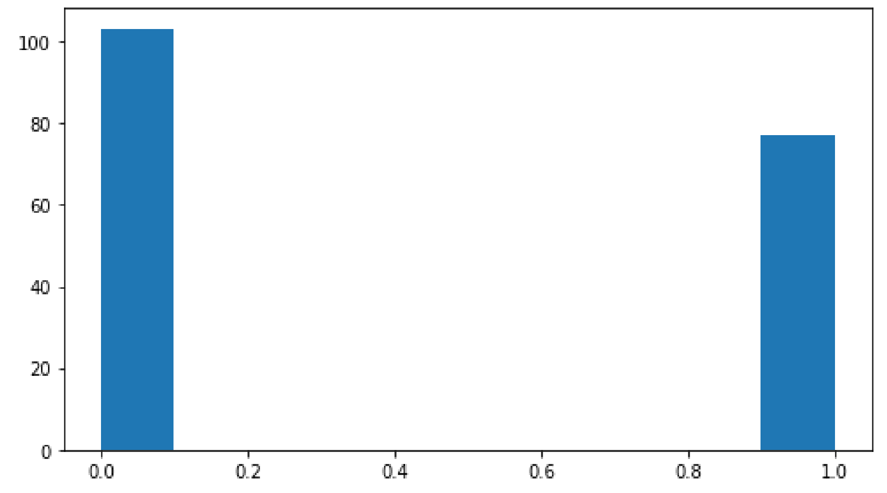
"Model accuracy is better with *SVM* than *NAIVE BAYS* algorithm"

Accuracy Scores

1. SVM =77.7 ~ 78%
2. Naive Bays = 73.8 ~ 74

```python
plt.hist(y_pred)
```

```
(array([103.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  77.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



Cross Validation

```
%cd /content/drive/MyDrive/Msc AI /TCS iON Rio -125 internship
!ls
```

```
/content/drive/MyDrive/Msc AI /TCS iON Rio -125 internship
 a1_RestaurantReviews_HistoricDump.csv  'Sentiment Analysis P1.ipynb'
 a2_RestaurantReviews_FreshDump.csv
```

```python
q = '/content/drive/MyDrive/Msc AI /TCS iON Rio -125 internship/a2_Restaur
```

```python
data = pd.read_csv(q)
data.head()
```

	Review
0	Spend your money elsewhere.
1	Their regular toasted bread was equally satisf...
2	The Buffet at Bellagio was far from what I ant...
3	And the drinks are WEAK, people!
4	-My order was not correct.

```python
data.count
```

```
<bound method DataFrame.count of                                               Review
0                         Spend your money elsewhere.
1   Their regular toasted bread was equally satisf...
2   The Buffet at Bellagio was far from what I ant...
3                    And the drinks are WEAK, people!
4                          -My order was not correct.
..                                                ...
95  I think food should have flavor and texture an...
96                           Appetite instantly gone.
97  Overall I was not impressed and would not go b...
98  The whole experience was underwhelming, and I ...
99  Then, as if I hadn't wasted enough of my life ...

[100 rows x 1 columns]>
```

```python
data.shape
```

```
(100, 1)
```

```python
data.describe
```

```
<bound method NDFrame.describe of                                               Review
0                         Spend your money elsewhere.
1   Their regular toasted bread was equally satisf...
2   The Buffet at Bellagio was far from what I ant...
3                    And the drinks are WEAK, people!
4                          -My order was not correct.
..                                                ...
95  I think food should have flavor and texture an...
96                           Appetite instantly gone.
97  Overall I was not impressed and would not go b...
98  The whole experience was underwhelming, and I ...
99  Then, as if I hadn't wasted enough of my life ...

[100 rows x 1 columns]>
```

```python
import re
import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()

all_stopwords = stopwords.words('english')
all_stopwords.remove('not')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```python
corpus=[]

for i in range(0, 100):
  review = re.sub('[^a-zA-Z]', ' ', data['Review'][i])
  review = review.lower()
  review = review.split()
  review = [ps.stem(word) for word in review if not word in set(all_stopwc
  review = ' '.join(review)
  corpus.append(review)
```

```python
# Loading BoW dictionary
from sklearn.feature_extraction.text import CountVectorizer
import pickle
cvFile='c1_BoW_Sentiment_Model.pkl'
# cv = CountVectorizer(decode_error="replace", vocabulary=pickle.load(open
cv = pickle.load(open(cvFile, "rb"))

```

```python
X_fresh = cv.transform(corpus).toarray()
X_fresh.shape
```

```
(100, 1420)
```

```python
import joblib
classifier = joblib.load('c2_Classifier_Sentiment_Model')
clf = joblib.load('Clf_Sentiment_Model_SVM')
```

```python
yf_pred = classifier.predict(X_fresh)
print(yf_pred)
```

```
[0 1 1 0 0 1 0 0 0 1 1 0 0 1 0 1 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 1 0
 0 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0]
```

```python
yf2_pred = clf.predict(X_fresh)
print(yf2_pred)
```

```
[0 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1
 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 0 0 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1
 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0]
```

```python
data['predicted_label'] = yf_pred.tolist()
data.head(10)
```

```
data['predicted_label'] = yf2_pred.tolist()
data.head(10)
```

```
data.to_csv('file1.csv',index= False)
```

```
len(y_test)
```

```
180
```

```
len(yf_pred)
```

```
100
```

```
len(yf2_pred)
```

```
100
```

Result Visulization

```
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('file1.csv')
```

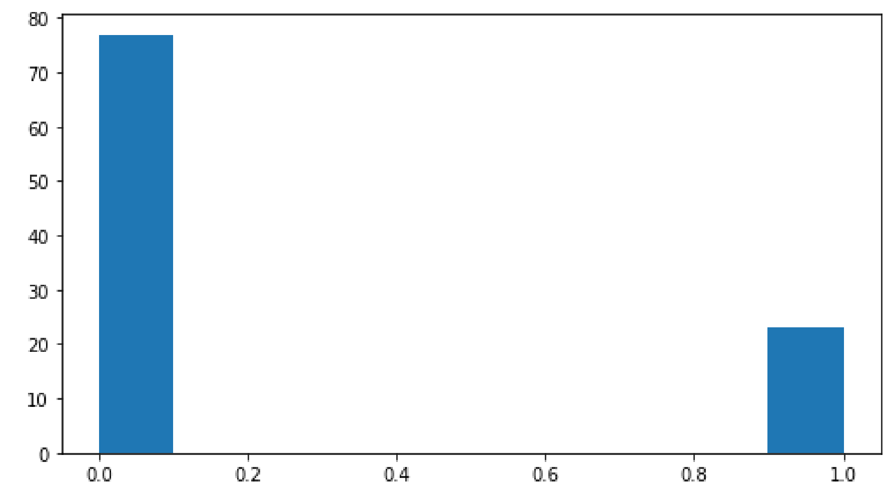
```
df.head()
```

```
plt.rcParams["figure.figsize"] = [7,4]
plt.rcParams["figure.autolayout"] = True
plt.title("Predicted values on unseen data")
# plt.hist(df['predicted_label'].head())
```

```
print(len(y_pred))
plt.title("y_pred viz",c='r')
plt.hist(y_pred)
```

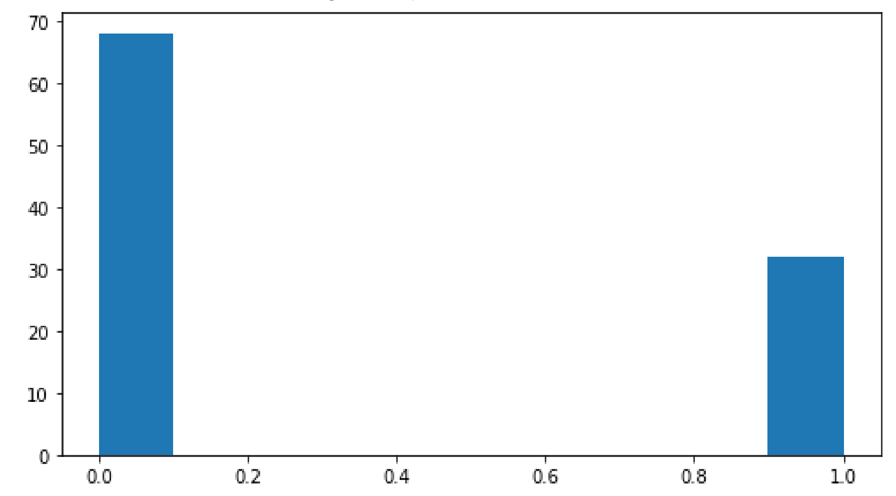
```
print(len(yf_pred))
plt.hist(yf_pred)
```

```
100
(array([77.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 23.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



```
print(len(yf2_pred))
plt.hist(yf2_pred)
```

```
100
(array([68.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0., 32.]),
 array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. ]),
 <a list of 10 Patch objects>)
```



We can conclud from the above plots that most of the feedbacks are negative feedbacks

End of The Project